

PLATINUM
technology



555 WatersEdge Drive
Lombard, IL 60148-9930
(708) 620-5000

Full
V2.3
Support

The Tools And Utilities For The DB2 Of Tomorrow. Tomorrow.

*available to thousands of locations across the U.S. †As of 12/91, support for DB2 V2.3 General Availability release of 10/2/91.
© 1992 PLATINUM technology, inc. All rights reserved. IBM and DB2 are registered trademarks and SystemView is a trademark
of International Business Machines Corporation. Federal Express and Priority Overnight are trademarks of Federal Express
Corporation and used with permission.

of the distributed database and makes the distribution transparent to users. We use the term distributed database system to refer to the combination of the distributed database and the distributed DBMS. The following assumptions about the system underlie these definitions:

□ Data is stored at a number of sites. Each site is assumed to consist logically of a single processor. Even if some sites are multiprocessor machines, the distributed DBMS is not concerned with the storage and management of data on these machines.

□ The processors at sites are interconnected by a computer network rather than a multiprocessor configuration. The important point here is the loose interconnection of processors that have their own operating systems and operate independently. Although "shared-nothing" multiprocessor architectures are similar to loosely interconnected distributed systems, they involve different issues (such as task allocation and migration and load balancing) that are not considered in this article.

□ The distributed database is indeed a database, not a collection of files that can be stored individually at each network node. This aspect marks a distinction between a distributed database and a dis-

Each site is assumed to consist of a single processor

tributed file system. To form a distributed database, distributed data should be logically related according to some structural formalism, and access to data should be at a high level via a common interface. The typical formalism establishing the logical relationship is the relational model. In fact, most research on distributed database systems assumes a relational system.

□ The system has the full functionality of a DBMS. It is neither a distributed file system nor a transaction-processing system. Transaction processing is not only one type of distributed application, but it is also among the functions provided by a distributed DBMS. However, a distributed DBMS provides other functions, such as query processing and structured data organization, which transaction-processing systems do not necessarily support.

These assumptions are valid in today's technology base. Most current distributed systems are built on local area networks where each

site is a single computer. The database is distributed so that each site manages a single local database (see Figure 1). Next-generation distributed DBMSs will be designed differently, however, as a result of technological developments—especially the emergence of affordable multiprocessors and high-speed networks. System design will also be affected by the increasing use of database technology in application domains that are more complex than business data processing and by the wider adoption of the client/server mode of computing, accompanied by the standardization of the client/server interface. Thus, the distributed DBMS environment will include multiprocessor database servers connected to high-speed networks linking them and other data repositories to client machines that run application code and participate in executing database requests. Distributed relational DBMSs of this type are already appearing, and several existing object-oriented systems also fit this description.

A distributed DBMS as defined here is only one way of providing database management support for a distributed computing environment. We have devised a working classification of possible design alternatives along three dimensions:⁴

□ *Autonomy* refers to control distribution and indicates the degree to which individual DBMSs can operate independently. It involves a number of factors, including whether the component systems exchange information, whether they can independently execute transactions, and whether a user is allowed to modify them. (In this context, "exchange information" does not refer to networking concerns but to whether the DBMSs are designed to exchange information and coordinate their actions in executing user requests.) Three types of autonomy are tight integration, semiautonomy, and full autonomy. In tightly integrated systems, a single image of the entire database is available to users who want to share the information in multiple databases. Semiautonomous systems consist of DBMSs that can (and usually do) operate independently but have been designed to participate in a federation to make their local data

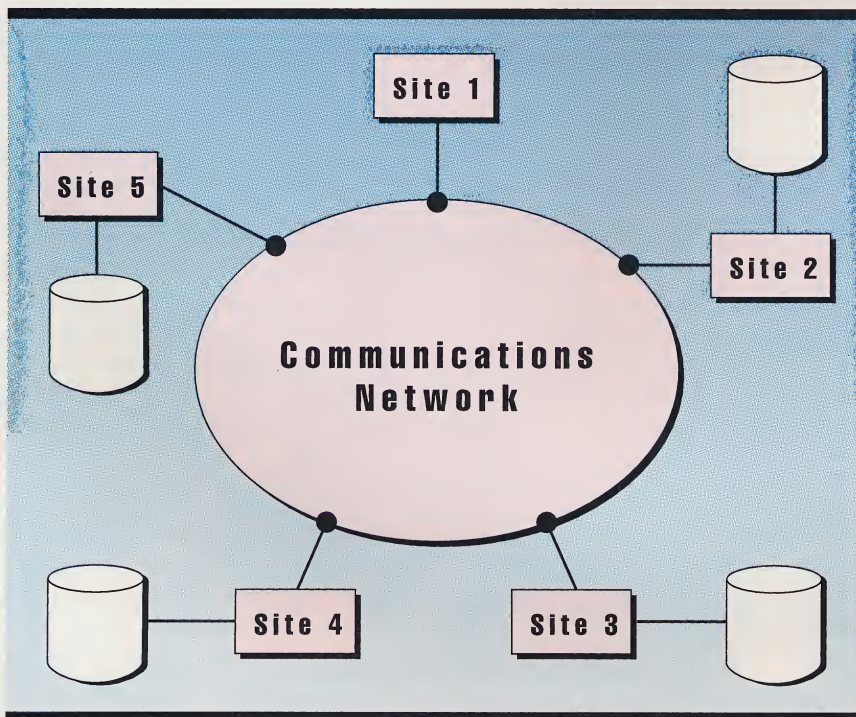


FIGURE 1. A distributed database environment.